FINITE WORD LENGTH EFFECTS IN DIGITAL FILTERS

Dr. Pooja sahni

Professor,ECE-CGC Landran

Fixed Point and Floating Point number Representations

Introduction

- Digital Signal Processing algorithms are realized either with special purpose hardware or general purpose digital computer.
- In both cases the numbers and coefficients are stored in finite length registers.
- The coefficients and numbers are quantized by truncation or rounding off when they are stored.
- The following errors arise due to quantization of numbers.
  - Input quantization error
  - Product quantization error
  - Coefficient quantization error

Number Systems

REVIEW OF NUMBER SYSTEMS

- It is a system in which quantities are expressed in numeric symbols.
- Numerous number systems are available.
- Knowledge of number system is the most essential requirement for understanding and designing the digital circuits.
- Generally number has 2 parts
    - Integer
    - Fractional, set apart by radix point

MOST COMMON NUMBER SYSTEMS

- Decimal system - (0 to 9) -------$()_{10}$
- Binary system – 0,1 ------$()_2$
- Octal system – (0 to 7) ---------$()_8$
- Hexadecimal system - 0 to 15 – 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
  - ----------$()_{16}$

Decimal system

- Most common number system used in day to day life.
- It is also known as base 10 system.

- Decimal to -------?------- (Binary or Octal or Hexadecimal)

- Binary to -------?------- (Decimal or Octal or Hexadecimal)

- Octal to -------?------- (Decimal or Binary or Hexadecimal)

- Hexadecimal to -------?------- (Decimal or Binary or Octal)

(Decimal) to (??)

Number system	Integer	Fractional
Dec to Bin	÷ 2	X 2
Dec to Octal	÷ 8	X 8
Dec to Hexadec	÷ 16	X 16

(Binary) to (??)

Number system	Integer	Fractional
Bin to Dec	X 2	÷ 2
Bin to Octal	Starting from Least Significant Bit, each group of 3 bits is replaced by its decimal equivalent	
Bin to Hexadec	Starting from Least Significant Bit, each group of 4 bits is replaced by its decimal equivalent	

(Octal) to (??)

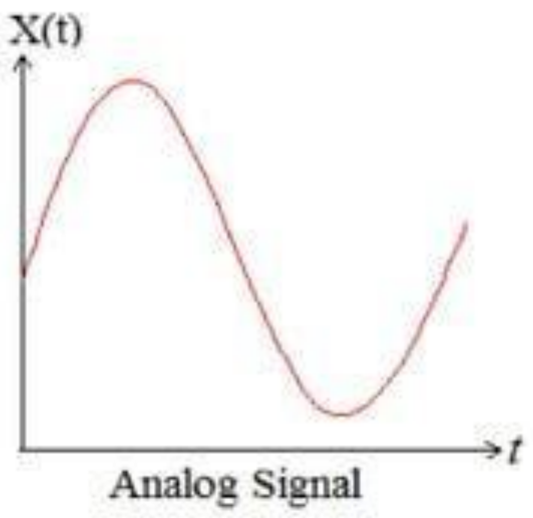
Number system	Integer	Fractional
Oct to Dec	X 8	÷ 8
Oct to Binay	Significant Octal digit is replaced by binary equivalent	
Oct to Hexadec	i). First Convert Octal to Binary ii). Then Binary to Hexadecimal	

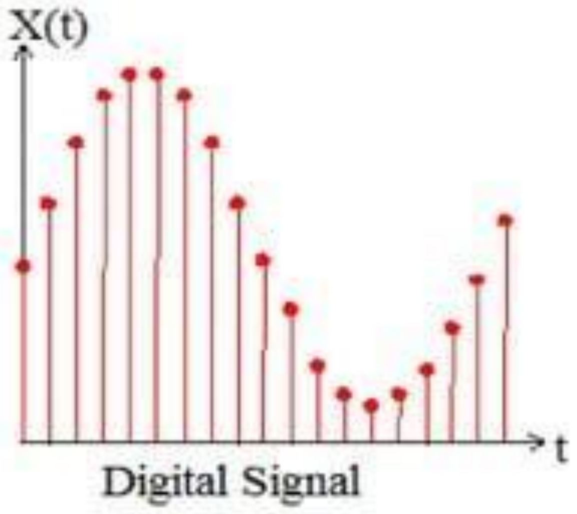
(Hexadecimal) to (??)

Number system	Integer	Fractional
Hexadec to Dec	X 16	÷ 16
Hexadec to Binay	Each Significant digit in the given number is replaced by its 4 bit binary equivalent	
Hexadec to Oct	i). First Convert Hexadec to Binary ii). Then Binary to Octal	

Analog and Digital Signal

- Analog system
  - The physical quantities or signals may vary continuously over a specified range.
- Digital system
  - The physical quantities or signals can assume only discrete values.
- Greater accuracy





Number System

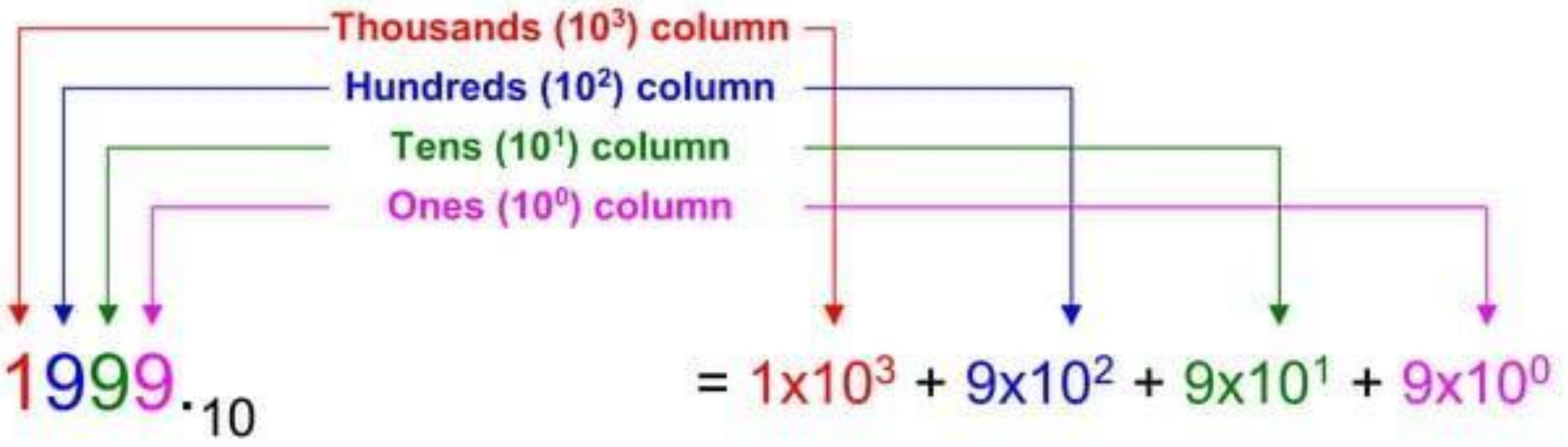
- The decimal number system is commonly used.
- Any number is possible to express in any base or radix "r".
- In general, any number with radix r, having m digits to the left and n digits to the right of the decimal point, can be expressed as

$$a_m r^{m-1} + a_{m-1} r^{m-2} + \dots + a_2 r^1 + a_1 r^0 . + b_1 r^{-1} + b_2 r^{-2} + \dots + b_n r^{-n}$$

- Where a_m is the digit in mth position.
- The coefficient a_m is termed as Most Significant Digit(MSD)
- b_n is termed as Least Significant Digit(LSD)

Base-10 (decimal) arithmetic

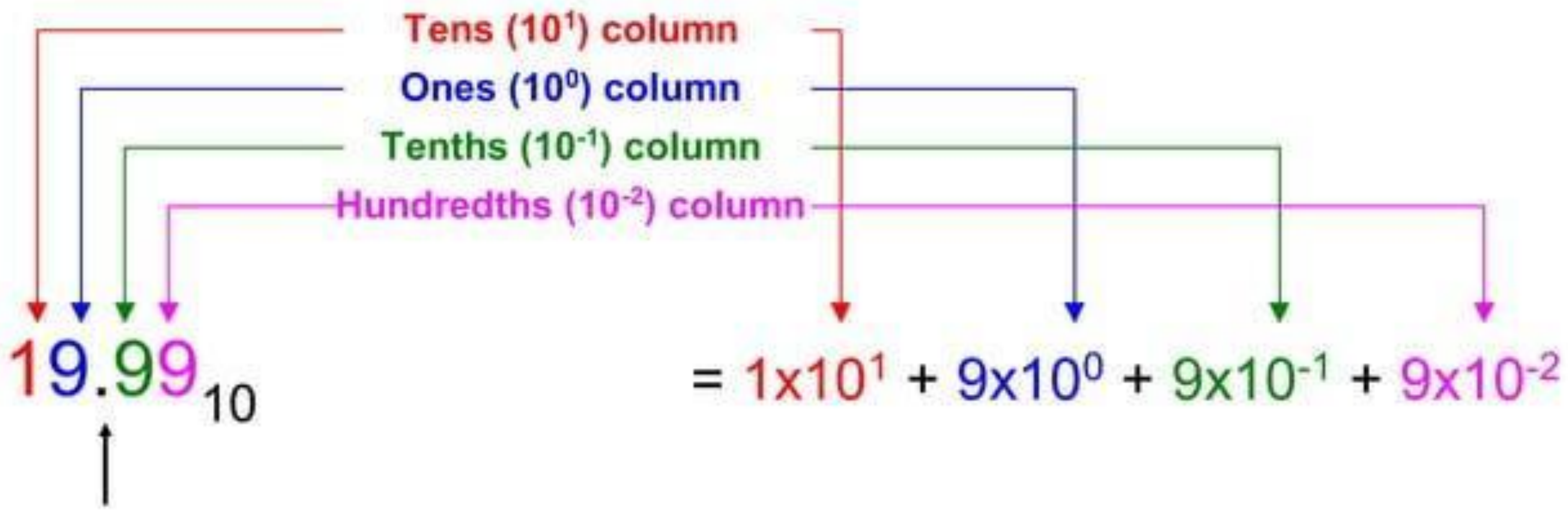
- Uses the ten numbers from 0 to 9
- Each column represents a power of 10



$$1999._{10} = 1\times10^3 + 9\times10^2 + 9\times10^1 + 9\times10^0$$

Base-10 (decimal) arithmetic

- Uses the ten numbers from 0 to 9
- Each column represents a power of 10



$$19.99_{10} = 1\times10^{1} + 9\times10^{0} + 9\times10^{-1} + 9\times10^{-2}$$

Addition

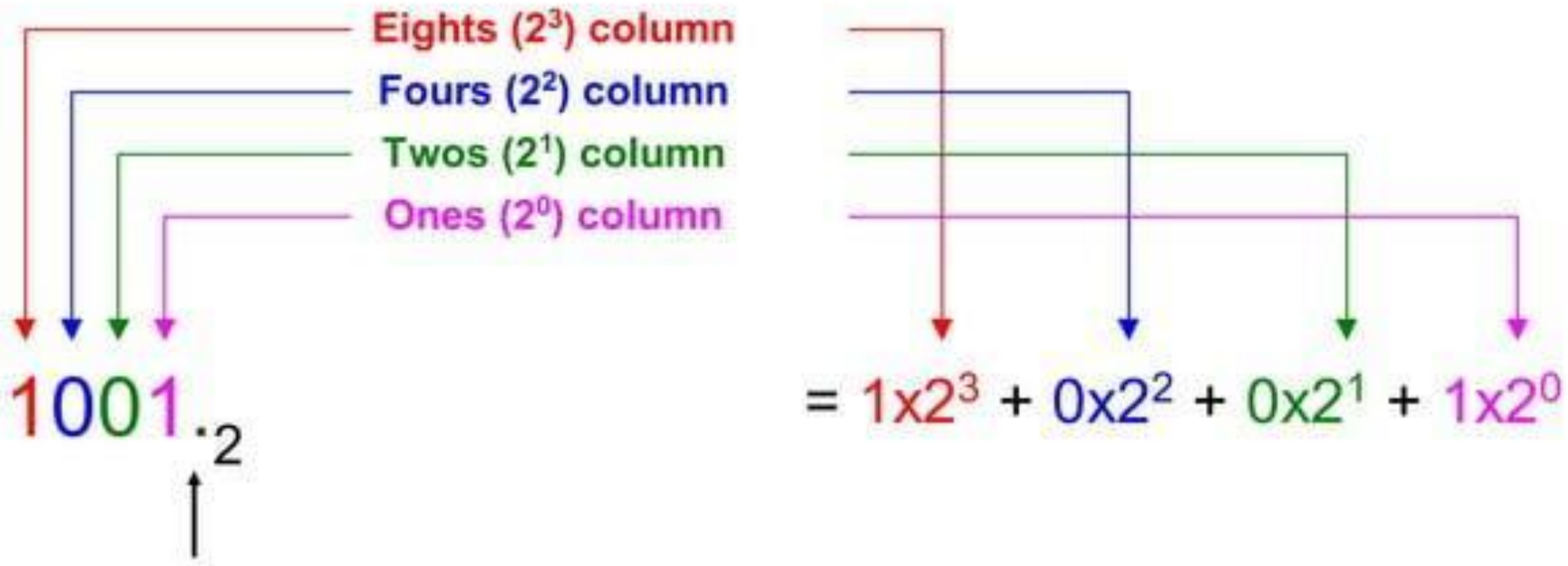
- Decimal Addition

$$\begin{array}{rccccl} & 1 & 1 & & \leftarrow & \text{Carry} \\ & & 5 & 5 & & \\ + & & 5 & 5 & & \\ \hline & 1 & 1 & 0 & & \end{array}$$

$$\searrow \; = \text{Ten} \geq \text{Base}$$

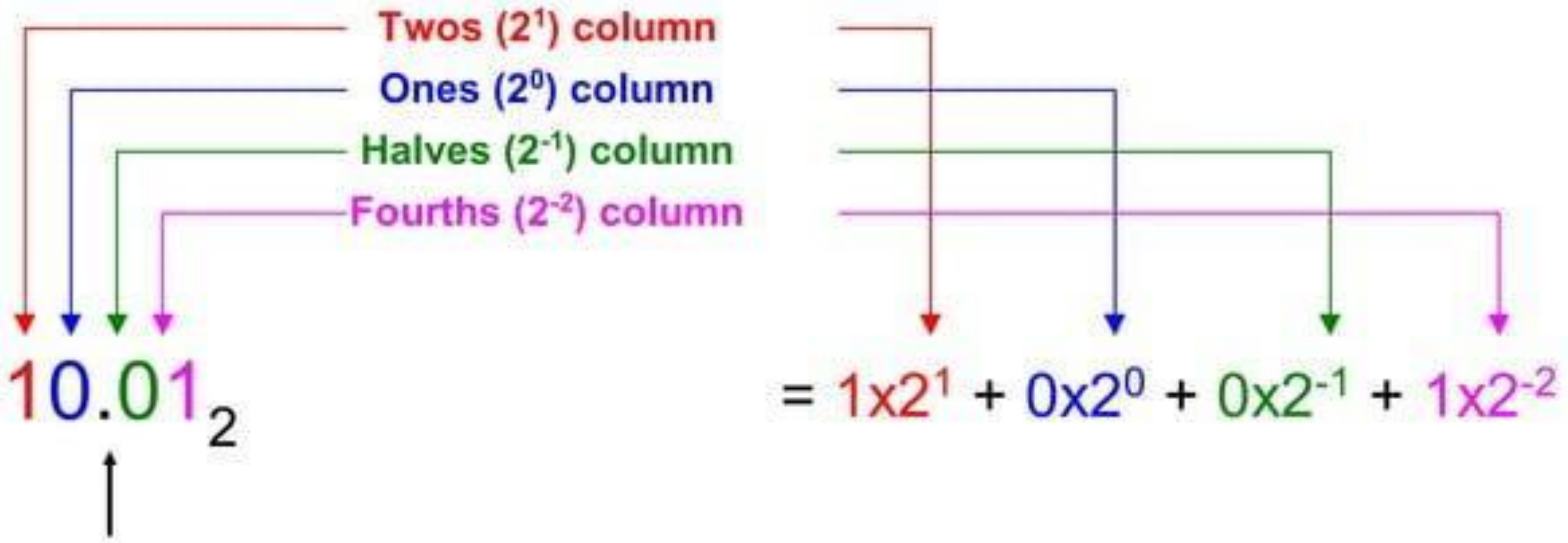
Standard binary representation

- Uses the two numbers from 0 to 1
- Every column represents a power of 2



Fixed-point representation

- Uses the two numbers from 0 to 1
- Every column represents a power of 2



10.01_2

$= 1 \times 2^1 + 0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$

Types of Number Representation

- There are 3 types
- These are used to represent the numbers in digital computer or any other digital hardware.

  - Fixed Point Representation
  - Floating Point Representation
  - Block Floating Point Representation

Fixed Point Representation

- In this arithmetic the position of the binary point is fixed.
- The bit to the right represent → Fractional part
- The bit to the left represent → Integer Part

(eg) The binary number 01.1100 has the value 1.75 in decimal

- The negative numbers are represented gives three different forms for fixed point arithmetic
  - Sign-Magnitude form
  - One's complement form
  - Two's complement form

Sign-Magnitude form

- The most significant digit is set to 1 to represent the negative sign.

(eg)

- The decimal number **-1.75** is represented as **11.110000**
- **1.75** is represented as **01.110000**

Signed Binary Numbers

- To represent negative integers, we need a notation for negative values.
- It is customary to represent the sign with a bit placed in the leftmost position of the number since binary digits.
- The convention is to make the sign bit 0 for positive and 1 for negative.
- Example:

Signed-magnitude representation:	10001001
Signed-1's-complement representation:	11110110
Signed-2's-complement representation:	11110111

Table 1 lists all possible four-bit signed binary numbers in the three representations.

Addition of two fixed point numbers

- The two numbers are added bit by bit starting from right, with carry bit being added to the next bit.

Eg. $(0.5)_{10} + (0.125)_{10}$

- Assuming the total number of bits b+1=4 (including sign bit)

we obtain

$$\begin{aligned}(0.5)_{10} &= 0.100_2\\ (0.125)_{10} &= \underline{0.001_2}\\ \text{sign bit} \longrightarrow\ & \underline{0.101_2} \longrightarrow (0.625)_{10}\end{aligned}$$

When two numbers of b bits are added and the sum cannot be represented by b bits an overflow is said to be occur eg.

$$
\begin{array}{rcl}
(0.5)_{10} & = & 0.100_2 \\
(0.625)_{10} & = & \underline{0.101_2} \\
\text{sign bit} \longrightarrow & & \underline{1.001_2} \longrightarrow (-0.125)_{10} \text{ in sign magnitude}
\end{array}
$$

- Overflow occurs in above result because $(1.125)_{10}$ cannot be represented in 3 bit number system.
- Thus in general, addition of fixed point numbers causes an overflow.
- The subtraction of two fixed point numbers can be performed easily by using two's complement representation.

Multiplication in fixed point arithmetic

- In Multiplication of two fixed point numbers first the sign and magnitude components are separated.
- The magnitude of the numbers are multiplied first, then the sign of the product is determined and applied to the result.
- When a b bit number is multiplied with another b bit number the product may contain 2b bits.

For eg. $(11)_2 \times (11)_2 = (1001)_2$.

- If b bits are organized into $b=b_i + b_j$

where b_i – Integer part & b_j – Fraction part

The product may contain $2b_i + 2b_j$ bits

In fixed point arithmetic, multiplication of two fraction results in a fraction. For multiplication with fractions overflow can never occur.

Floating Point representation

- In floating point representation a positive number is represented as

$$F=2^{C}.M$$

- where M → mantissa is a fraction such that $\frac{1}{2} \leq M \leq 1$
- C → exponent , either positive or negative.

- The decimal numbers 4.5, 1.5, 6.5 and 0.625 have floating point representations as

$2^3 x0.5625$, $2^1 x0.75$, $2^3 x0.8125$, $2^0 x0.625$ respectively

- Equivalently

$$2^3 x0.5625 = 2^{011} x0.1001$$
$$2^1 x0.75 = 2^{001} x\ 0.1100$$
$$2^3 x0.8125 = 2^{011} x0.1101$$
$$2^0 x0.625 = 2^{000} x0.1010$$

$$(3)_{10} = (011)_2$$
$$(0.5625)_{10} = (0.1001)_2$$
$$(1)_{10} = (001)_2$$
$$(0.75)_{10} = (0.1100)_2$$
$$(0.8125)_{10} = (0.1101)_2$$
$$(0.625)_{10} = (0.1010)_2$$

- Negative floating point numbers are generally represented by considering the mantissa as a fixed point number.
- The sign of the floating point number is obtained from the first bit of mantissa.
- The floating point arithmetic multiplications are carried out as follows. Let $F_1=2^{C1}xM_1$ and $F_2=2^{C2}xM_2$
- Then the product $F_3=F_1 \times F_2 = (M_1 \times M_2)\, 2^{C1+C2}$

(i.e) the mantissa are multiplied using fixed point arithmetic and exponents are added.

The product ($M_1 \times M_2$) must be in the range of 0.25 to 1.0.

To correct this problem the exponent (C1+C2) must be altered.

- $(1.5)_{10} = 2^1 \times 0.75 \quad = 2^{001} \times 0.1100$
- $(1.25)_{10} = 2^1 \times 0.625 = 2^{001} \times 0.1010$
- $(1.5)_{10} \times (1.25)_{10} \quad = (2^{001} \times 0.1100) \times (2^{001} \times 0.1010)$

  $= 2^{010} \times 0.01111$

$(0.75)_{10} = (0.1100)_2$
$(0.625)_{10} = (0.1010)_2$
$\underline{(0.1100)_2 \times (0.1010)_2}$
$\underline{(0.01111000)_2}$

- Addition and subtraction of two floating point numbers are more difficult than the addition and subtraction of two fixed point numbers.

To carry out addition, first adjust the exponent of the smaller number until it matches the exponent of the larger number. the mantissa are then added and subtracted.

Finally, the resulting representation is rescaled so that its mantissa lies in the range 0.5 to 1.

- Suppose we are adding 3.0 and 0.125

$$3.0=2^{010} \times 0.11000$$

$$0.125=2^{000} \times 0.001000$$

- Now we adjust the exponent of smaller number. So that both exponents are equal.

$$0.125=2^{010} \times 0.0000100$$

- Now the sum is equal to $2^{010} \times 0.110010$

Fixed Point Arithmetic	Floating point Arithmetic
• Fast Operation	• Slow Operation
• Relatively economical	• More Expensive because of costlier hardware
• Small dynamic range	• Increased dynamic range
• Round off Errors occur only for addition	• Roundoff errors can occur with both addition and multiplication
• Overflow occur in addition	• Overflow does not arise.
• Used in small computers	• Used in larger, general purpose computers.

Thank you